Reinforcement Learning of Phase Oscillators for Fast Adaptation to Moving Targets

Guilherme Maeda[1], **Okan Koc**[2], **Jun Morimoto**[1]
[1]Department of Brain Robot Interface, ATR Computational Neuroscience Laboratories, Japan
[2]Department for Empirical Inference and Machine Learning, Max-Planck Institute, Germany
g.maeda@atr.jp, okan.koc@tuebingen.mpg.de, xmorimo@atr.jp

Abstract: Online movement generation in tasks involving real humanoid robots interacting with fast-moving targets is extremely difficult. This paper approaches this problem via imitation and reinforcement learning using phase variables. Imitation learning is used to acquire primitive trajectories of the demonstrator interacting with the target. The temporal progress of the robot is represented as a function of the target's phase. Using a phase oscillator formulation, reinforcement learning optimizes a temporal policy such that the robot can quickly react to large/unexpected changes in the target movement. The phase representation decouples the temporal and spatial problems allowing the use of fast online solutions. The methodology is applicable in both cyclic and single-stroke movements. We applied the proposed method on a real bi-manual humanoid upper body with 14 degrees-of-freedom where the robot had to repeatedly push a ball hanging in front of it. In simulation, we show a human-robot interaction scenario where the robot changed its role from giver to receiver as a function of the interaction reward.

Keywords: Policy search, imitation learning, phase oscillator

1 Introduction

Programming robots to achieve highly adaptive motor skills such as the ones required to interact with a fast-moving object presents a number of challenges. The required computational time to generate and adapt the robot's trajectories is one of the biggest problems. Humanoid robots that mimic the human kinematics present many degrees-of-freedom (DoFs) which makes online generation of the required trajectories extremely difficult. For example, Bäuml et al. [1] reports a humanoid on a mobile base used for catching flying balls which required a dedicated cluster with 32 cores to solve trajectory generation routines at a frequency of 25 Hz. Early works in fast dynamic manipulation [2, 3] recurred to strong assumptions on the path described by the target to simplify computations. While recent works have explored approaches such as fast visual processing [4], careful parameterization of trajectories, and modeling of object dynamics [5]; the online nature of the task is still computationally challenging and requires great engineering effort.

Movement primitives have proven extremely successful in fast, dynamic tasks. Primitives not only decrease the dimensionality of the movement representation but also facilitate their generalization to new situations. Compared to optimal trajectory generation, primitives can be learned by imitation making them easy to program and apply in a variety of tasks. In robotics, Dynamical Movement Primitives (DMPs) [6] are perhaps the most popular formulation of movement primitives. They have been successfully used in a variety of highly dynamical manipulation tasks [7, 8, 9], but only indirectly connected to the target evolution. As an example, in [7], the time and velocity of the hitting point of a ball were first computed using flight model predictions whose values were then fed as parameters to adapt the DMPs. In this paper, we present a solution where fast reactive skills are achieved by primitives that can be directly driven by an observed target.

One approach to enforce a tighter connection between the target and robot movements is to use the phase of the former to govern the progress of the latter. The phase is an indicator of the progress of

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

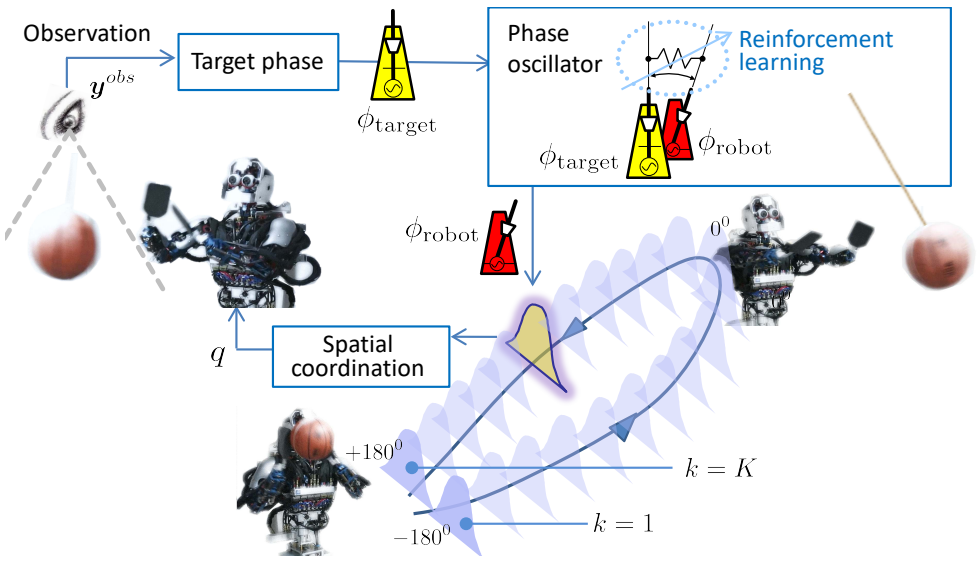

Figure 1: The main components of the method for a periodic task. Observations $\boldsymbol{y}^{obs}$ are used to estimate the phase of the target ϕ_{target}. The oscillator computes the robot's phase ϕ_{robot} as a function of ϕ_{target}. The phase is used to select the corresponding spatial model to retrieve appropriate robot joint angles $\boldsymbol{q}$. Reinforcement learning plays a fundamental role during training in order to adapt/refine the relationship between the phases. A properly specified reward and successful optimization can lead to complex and robust temporal interactions beyond the reach of a manually tuned oscillator. Note that the oscillator can advance ϕ_{robot} with respect to ϕ_{target} for predictive robot actions.

an agent/object on a task as a function of its state rather than time. By leveraging on demonstrations and models learned by imitation, it is possible to estimate the phase trajectory of the target such that the robot can anticipate actions ahead of the current state of the task.

Movement primitives indexed by phase variables have been widely used in locomotion [10, 11, 12, 13]. In particular, phase oscillators have been used to add structure and suitably constrain the temporal freedom of multi-DoF systems. They have shown useful, for example, to speed up the learning of a walking pattern in a humanoid [12]. While in locomotion it is common to use an internal phase source to provide an inner clock for multiple joints, a less explored idea is to use the phase of an external source to govern the progress of the robot (e.g., to juggle a ball [14]). Here, we will leverage on the external clock insight to solve highly dynamical manipulation tasks in both rhythmic and discrete movements.

The use of phase indexes solves a great part of the control problem as it temporally locates the robot progress with respect to a target. However, at each of these indexes, appropriate robot poses still need to be computed as a function of the target's spatial location. This brings the problem of designing spatial models. In tasks such as bi-manual manipulation of a ball, computing robot poses online can be difficult due to the unknown interaction forces and the geometry of the contact points. Thus, we use expert demonstrations to record the Cartesian trajectories of a demonstrator's hands and the target as he/she manipulates it. Using a retargeting procedure, the extracted relative positions of the hands and the object provide a deterministic backbone from which spatial correlations can be added during training. In physical human-robot interaction, the same arguments hold if the target is, for example, the position of another person's hand.

The principal contribution of this paper is an online algorithm for fast temporal and spatial adaptation to a moving target by modulating the behavior of a phase oscillator. We introduce policy search RL as a means to automatically adapt the phase oscillator parameters and investigate how to efficiently design spatial models that work in concert with the phase formulation. Due to the low computational load, the method is scalable and can be used in both cyclic and single-stroke movements. We tested

the method on a real humanoid upper body with 14 DoFs manipulating a fast-moving ball, and addressed in simulation a scenario of physical human-robot interaction.

2 Learning Spatial and Temporal Coordination for Fast Adaptation

Figure 1 schematically shows the overall methodology. To provide a concrete application, consider a robot that must block an incoming ball and immediately strike it as far as possible. The ball is returned after each strike by another player or because it could be attached to the ceiling by a rope. At each control cycle, the current position of the target is observed $\boldsymbol{y}^{obs}$, from which a target phase ϕ_{target} is computed. Given the rhythmic nature of this particular example, one can interpret the phenomena generating the phase as a metronome. A phase oscillator couples the target's metronome with the robot's metronome via feedback. The output of the oscillator is the robot phase ϕ_{robot}, which is used to index a sequence of K pre-trained spatial models $P(\boldsymbol{q}, \boldsymbol{y})_{1:K}$. By using a probabilistic formulation, these models can be conditioned on the target position to output robot joint angles.

Reinforcement learning (RL) is used to modulate the nonlinear policy that connects the two metronomes. For example, a reward that penalizes the robot for not moving the ball will enforce the robot to put the ball on a limit cycle. Conversely, a reward that penalizes the ball movement will make the robot to stop the ball without a subsequent push. Optimizing phase policies under different rewards opens the possibility to train the robot for a variety of behaviors whose governing characteristic is temporal rather than spatial. For example, simulations in Section 3.1 show that the robot can behave either as a giver or receiver in handover tasks. Also, RL allows us to refine the temporal coordination between the robot and the moving target. For single-stroke movements, the only difference in the formulation is that the phase is defined in a non-repetitive interval (e.g. [0, 180^0]); back to our analogy, the needles of the metronomes move only once from left to right and stop. As shall be clear by the end of this section, computing the phase ϕ_{robot} and the joint angles $\mathbf{q}$ at each time step basically involves integrating a differential equation (5), and multiplying matrices of dimensions no larger than the number of robot joints (4). As such, these operations can be carried out online at high frequencies.

2.1 Phase-Indexed Spatial Models

The whole sequence of phase-indexed spatial models in Figure 1 constitutes a movement primitive. Each model is a joint distribution of the form

$$P(\boldsymbol{q}, \boldsymbol{y})_k \text{ with } k \in 1, ..., K, \tag{1}$$

where k is the robot phase index, $\boldsymbol{q} = (q_1, q_2, ...q_J)_k^\top, j \in \{1, ..., J\}$ is a vector containing reference positions for each of the joints of the robot, and $\boldsymbol{y} = (x, y, z)_k^\top \in \mathbb{R}^3$ are the observed Cartesian coordinates of the target at the k-th index. Note that on a cyclic (rhythmic) task, the sequence (1) must form a closed path.

During training, for each phase index k, assume N pairs of target positions and corresponding robot joint angles are provided; that is, $(\boldsymbol{q}_n, \boldsymbol{y}_n)_k \in \mathbb{R}^{(J+3)}$ is one of the N training samples as a vector formed by the concatenation of the joint angles of the robot with the corresponding target location. Under a normal distribution

$$\boldsymbol{\mu}_{q,y} = \text{mean}([(\boldsymbol{q}_1, \boldsymbol{y}_1)_k, ...(\boldsymbol{q}_N, \boldsymbol{y}_N)_k]^\top) = [\boldsymbol{\mu}_q, \ \ \boldsymbol{\mu}_y]_k^\top, \tag{2}$$

$$\boldsymbol{\Sigma}_{q,y} = \text{cov}([(\boldsymbol{q}_1, \boldsymbol{y}_1)_k, ...(\boldsymbol{q}_N, \boldsymbol{y}_N)_k]^\top) = \begin{bmatrix} \boldsymbol{\Sigma}_{qq} & \boldsymbol{\Sigma}_{qy} \\ \boldsymbol{\Sigma}_{yq} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}_k. \tag{3}$$

The primitive is then a sequence $P(\boldsymbol{q}, \boldsymbol{y})_{1:K} = \mathcal{N}(\boldsymbol{\mu}_{q,y}, \boldsymbol{\Sigma}_{q,y})_{1:K}$.

At a given phase index k, the conditional distribution $P(\boldsymbol{q}|\boldsymbol{y}^{obs})_k = \mathcal{N}(\boldsymbol{\mu}_{q|y}, \boldsymbol{\Sigma}_{q|y})_k$ is used to infer robot joint angles given the target observation. The conditional is computed in closed form

$$\begin{aligned} \boldsymbol{\mu}_{q|y} &= \boldsymbol{\mu}_q + \boldsymbol{\Sigma}_{qy}\boldsymbol{\Sigma}_{yy}^{-1}(\boldsymbol{y}^{obs} - \boldsymbol{\mu}_y) \\ \boldsymbol{\Sigma}_{q|y} &= \boldsymbol{\Sigma}_{qq} - \boldsymbol{\Sigma}_{qy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yq}. \end{aligned} \tag{4}$$

Note that except for the innovation $\boldsymbol{y}^{obs} - \boldsymbol{\mu}_y$, all other terms can be pre-computed and stored in advance.

The process of collecting training data, that is, the N pairs $(\boldsymbol{q}_{1:N}, \boldsymbol{y}_{1:N})_k$ at each of the k steps along a trajectory differs from task to task. As in many probabilistic approaches to trajectory modeling, this process usually involves multiple expert demonstrations of paired robot-target trajectories (e.g. by kinesthetic teaching). These demonstrations are then time-aligned, and for the case of the method here presented, discretized in K time slices. In the Appendix, we describe a heuristic for sampling numerous target positions and robot poses using a single demonstration and a kinematic model of the scene.

2.2 Modulating Phase Oscillators via Policy Search

So far we have K trained joint probability distributions $P(\boldsymbol{q}, \boldsymbol{y})_{1:K}$ where the index k is given by the robot phase ϕ_{robot}. Cyclic tasks are defined by a one-to-one mapping between the indexes $k = \{1, ..., K\}$ and the phase $\phi = \{-180^0, ..., 180^0\}$. For single-stroke cases, the phase is defined to be a sequence $\phi = \{0^0, ..., 180^0\}$. The proposed method couples the phase of the robot to the phase of the target with an oscillator of form [14, 10]

$$\dot{\phi}_{\text{robot}} = \omega + K \sin(\phi_{\text{target}}(y, \dot{y}) - \phi_{\text{robot}} + \alpha), \tag{5}$$

where the first term ω is the natural frequency of the robot movement and paces the rhythm of its cycle if no corrections are made. The second term is a correction that attempts to lock ϕ_{robot} with ϕ_{target} with an offset $\alpha = \phi_{\text{target}} - \phi_{\text{robot}}$, thus similar to a phase impedance. The feedback gain K defines how aggressively the tracking of the phase lock is made. The phase of the external target is computed as in [15] as

$$\phi_{\text{target}} = -\arctan\left(\frac{\dot{y}}{y}\right), \tag{6}$$

where $y,\ \dot{y}$ are the position and velocity of the target, respectively.

Reinforcement learning is used to tune the nonlinear impedance of the oscillator via policy search. Assuming the impedance is governed by the state of the external variable, we define the stiffness between phases and the phase difference as a function of the target phase, $K(\phi_{\text{target}}(y, \dot{y}))$ and $\alpha(\phi_{\text{target}}(y, \dot{y}))$ such that (5) becomes

$$\dot{\phi}_{\text{robot}} = \omega + K(\phi_{\text{target}}(y, \dot{y})) \sin[\phi_{\text{target}}(y, \dot{y}) - \phi_{\text{robot}} + \alpha(\phi_{\text{target}}(y, \dot{y}))], \tag{7}$$

As usual in policy search, the functions $K(\phi_{\text{target}}(y, \dot{y}))$ and $\alpha(\phi_{\text{target}}(y, \dot{y}))$ are first parameterized via linear regression on radial-basis functions. Thus, the search is done in the parameters space, which has a lower dimensionality and produces smooth policies. We define the vector of parameters

$$\boldsymbol{\theta} = [\boldsymbol{\theta}_K,\ \boldsymbol{\theta}_\alpha] \tag{8}$$

where,

$$\boldsymbol{\theta}_K = \left(\boldsymbol{\Phi}_K^\top \boldsymbol{\Phi}_K\right)^{-1} \boldsymbol{\Phi}_K^\top K(\phi_{\text{target}}) \text{ and } \boldsymbol{\theta}_\alpha = \left(\boldsymbol{\Phi}_\alpha^\top \boldsymbol{\Phi}_\alpha\right)^{-1} \boldsymbol{\Phi}_\alpha^\top \alpha(\phi_{\text{target}}), \tag{9}$$

and $\boldsymbol{\Phi}_{(\cdot)}$ are design matrices with radial-basis functions with centers evenly spread over the phase interval. A policy search philosophy is attractive due to its sampling efficiency in real robotic tasks [16]. Here, we use updates based on reward re-weighting as in PoWER[17], Pi^2[18] and Pi$^{\text{BB}}$ [19] in an episodic setting

$$\boldsymbol{\theta}^{new} \leftarrow \boldsymbol{\theta} + \sum_{m=1}^{M} [P(\tau_m)\epsilon_m], \tag{10}$$

where M is the number of rollouts obtained by adding noise ϵ_m from a pre-defined distribution to produce exploratory policies τ_m. The cost $S(\tau_m)$ of each rollout is used to score the perturbation ϵ_m with a weight given by a soft-max distribution [18, 19]

$$P(\tau_m) = \frac{\exp[-\lambda^{-1} S(\tau_m)]}{\sum_{m'=1}^{M} \exp[-\lambda^{-1} S(\tau_m)]}, \tag{11}$$

where the scalar λ is a user-defined temperature. The method here proposed is not linked to a particular policy search implementation. In this paper we used a general black-box formulation based on Pi$^{\text{BB}}$ [19] but, in principle, other policy search methods such as the ones reviewed in [19, 16] are also applicable.

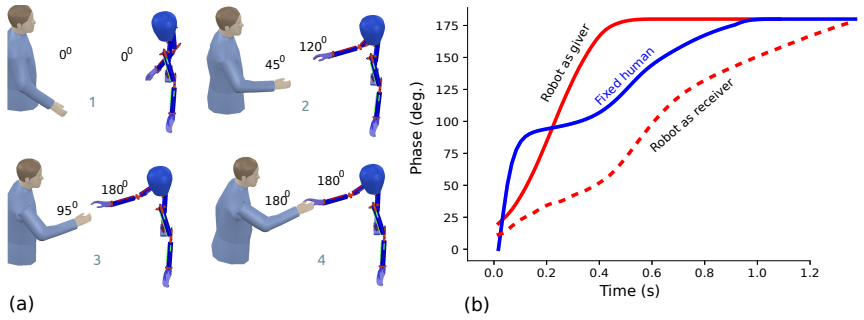

Figure 2: Single-stroke (non-cyclic/discrete) phase coordination in a handover task. The figures on the left illustrate a sequence of snapshots and the respective phases of the human partner and the robot. In this particular example, the robot hand arrives at the handover location before the hand of the partner. The figure on the right shows two cases where the robot acts as a giver and receiver roles w.r.t. the fixed trajectory of the human. These two robot behaviors are the result of a simple change in sign of the reward.

3 Experiments with a Humanoid Robot

This section reports experimental and simulated results using the upper body of a human-sized hydraulic humanoid robot [20] in both single-stroke and cyclic cases.

3.1 Policy Search in Simulation

Human-Robot Interaction. In simulation, we first evaluated the non-cyclic human-robot handover task to verify the method's capability to change the temporal behavior of the robot. The phase indexes were constrained within the interval $[0, 180^0]$ such that at $\phi = 0^0$ both agents are at their initial positions (as shown in Figure 2(a) snapshot 1). The simulated robot has the same kinematics of the hydraulic humanoid used for the real ball pushing experiments. The simulated human partner is a kinematic model provided by the simulation environment [21]. The hand's trajectories were obtained via demonstration of a two-person handover. We retargeted the two demonstrators, one to each agent in the scene and fixed the trajectory of the simulated human partner (the blue curve in Figure 2(b)). The moving target is defined to be the hand of the simulated human. Since this simulation is concerned with the evaluation of the temporal coordination only, the trajectories of the simulated agents are deterministic such that the probabilistic spatial models of Section 2.1 contain only the mean[1].

The reward was defined as having a single component where we measured the time each of the agents took to reach $\phi_{(\cdot)} = 180^0$, which corresponds to the moment they reach the end of their respective trajectories (snapshot 4 in Figure 2). Thus, by defining

$$T_{(\cdot)} = t(\phi_{(\cdot)} = 180^0)$$

as the final time, we can, for example, enforce the difference $\Delta T = T_{\text{target}} - T_r$ to be zero such that both hands arrive at their final location at the same time as in a handshake. If ΔT is positive, the human takes longer than the robot to arrive at the handover location, such that the robot acts as a giver. If ΔT is negative, the roles are reversed.

The blue curve in Figure 2(b) shows the progress of the fixed human movement as a function of time, which is a direct reproduction of one of the demonstrators. The solid red curve shows the phase evolution of the robot as a giver, by specifying a cost as $S = |\Delta T - 0.3|$ such that the robot is enforced to arrived 0.3 seconds before the human. This means that once the robot started observing the human motion, it moved slightly faster than her, offering the robot the opportunity to impose where to handover the object. The dashed red curve shows the phase of the robot for the same human movement, but now with a cost $S = |\Delta T + 0.3|$, which is suitable when the robot is the receiver. Note that these non-trivial changes of robot behavior become straightforward with the use of phases.

[1]We are currently investigating how to obtain samples for the handover task without relying on extensive demonstrations so that the spatial coordination problem can be addressed in later work.

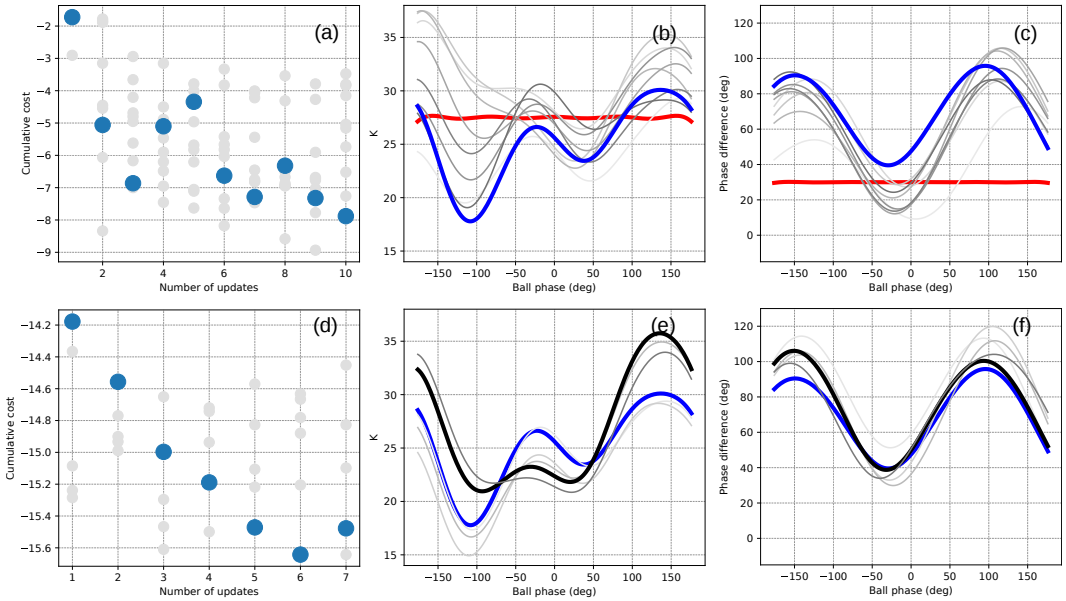

Figure 3: Cost and policy improvements in reinforcement learning. The top row shows the initial learning using the simulator. The bottom row shows the learning using the real robot which started with the results from the simulator. From (c) and (f) it is noted that the final phase difference did not change significantly, however, the gain difference between (b) and (e) increased after the optimization on the real system.

Pre-training an Initial Policy for Ball Pushing Skills. In the cyclic ball pushing task the two arms of the robot are used, totaling 14 DoFs to be controlled. The ball was attached to the ceiling by a rope of 1.5 meters and its position was captured at a frequency of 30 Hz by a Kinect2 camera[2]. As it was motivated in Figure 1, the goal of this task was to make the robot generate a limit cycle for the ball. Our initial attempts to use fixed hand-tuned parameters on the oscillator were unsuccessful; the robot either became good at receiving the ball, or good at pushing it, but never good at both at the same time. Also, softly decelerating the incoming ball to avoid bouncing suggests a smooth phase tracking of the ball. Conversely, to forcefully push the ball away, the robot must aggressively attempt a phase lock, which demands high gains. Simultaneously tuning these functions to have the required nonlinear profiles by hand proved impractical.

Recurring to RL to optimize the oscillator, we defined an immediate cost for the i-th time step of a rollout as

$$s_i = 10\,\mathrm{MS}(\ddot{q}_{1:14}) + 5\,|y_{\text{goal}} - y_{\text{ball}}| - 20(0.5|y_{\text{left}} - y_{\text{ball}}| + 0.5|y_{\text{right}} - y_{\text{ball}}|) \tag{12}$$

such that a rollout with N time steps has a cost $S = \sum_{i=1}^{N} s_i/N$. The first term in (12) is the mean squared of all joint accelerations (7 for each arm) in rad/s^2. The second term is a cost that enforces the robot to push the ball far away from its origin, in this case by a distance of $y_{\text{goal}} = 3$ meters. The third term is an average of the left and right hands' rewards that is maximized by keeping the distance between the robot's left and right paddles (y_{left}, y_{right}, respectively) away from the current ball position.

An intrinsic problem of using RL on real robots is that initial aggressive exploratory policies can damage the robot but at the same time are often essential to find good final solutions. Thus, a simulator with the robot kinematics and ball dynamics was used to start the policy search. Figure 3(a) shows the simulated cost reduction according to the number of updates. The bigger blue circles represent the cost of the noiseless rollout of the current policy (i.e. $\epsilon_m = 0$), and the gray markers show the cost of each noisy rollouts ($\epsilon_m \neq 0$). Figures 3(b-c) show the optimized robot policy for the phase difference α and the coupling stiffness K. As indicated by the red curves, the initial phase

[2]The camera is the bottleneck of the frequency in which joint angles can be computed. The computation time to generate new robot joint angles given a new ball measurement is usually one to two orders of magnitude faster than the camera frequency.

difference was set to 30° and the coupling gain to 27.5 over the whole phase range. The blue curves indicate the final optimized policies and the gray curves represent the intermediate clean rollouts of each intermediate updates. The simulation was comprised of 10 policy updates, where each update required a set comprised of one clean and nine noisy rollouts. The equivalent duration of the training process would be of 50 minutes on the real robot (not including the time to reset the ball and robot positions after each iteration).

3.2 Policy Search on the Real Ball Pushing Scenario

To refine the policy using the real robot, the ball was set such that when resting it was approximately 60 cm away from the robot paddles, giving the ball plenty of room to swing. Each rollout was set to take 30 seconds. Since the cost (12) favors interactions in which the ball is far from the paddles, good policies eventually put the system on a regime of persistent oscillation. Figure 3(d) shows the policy costs using the real robot. In (e-f), the initial policy coarsely optimized in simulation is shown in blue, and the final policy is shown in black. Using the real robot, each update comprised 5 rollouts and they were run until the cost eventually increased, totaling 35 rollouts and 17 minutes of effective interaction with the ball. The cost of the real case was lower than in simulation, as the ball tended to move farther in the real world.

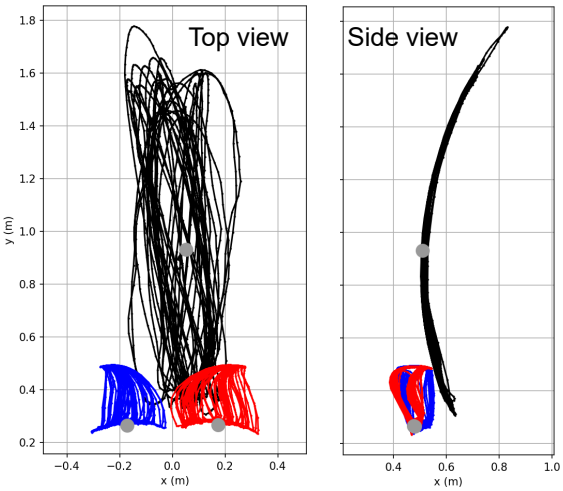

Figure 4: Experimental results using the optimized policy where the robot interacted with the ball for one minute. The blue, red and black curves show the trajectories of the left paddle, the right paddle, and the ball, respectively. Note from the side view the arc-shaped trajectory as the ball was attached to the ceiling by a 1.5 meter rope.

Comparing Figures 3(b) and (e), note that the coupling gain between phases increased substantially when using the real robot. This increase suggests that RL is tuning the policy to compensate for the dynamics of the real robot, which has delays, tracking error, and compliance—while in simulation a perfect kinematic robot with infinite bandwidth was used. Also, note from (e) and (f) that the gains and phase differences tend to have higher values at the beginning of the cycle. In fact, the pushing occurs around -180^0 to -130^0, and as expected, it requires a large phase difference as it will put the motion of the robot ahead of the ball. The gain is high, which improves the phase tracking at this stage. Around the middle of the cycle (when the ball phase is zero) the ball is usually far away from the robot's reach, so the gain and phases difference are of less importance. As the ball returns to the robot (between 50-150 degrees), the phase difference increases again such that the robot paddles return faster than the ball, and the difference decreases at the end, an indication that the robot attempts to approximate its phase with the ball phase for a synchronized deceleration.

3.3 Evaluating the Optimized Policy

Figure 4 shows real-robot results using the optimized policy where the robot pushed the ball for an entire minute. The ball and robot hand trajectories are shown from the top and from the side. The strongest pushes could move the ball more than 1.5 meters away. The swing of the ball barely stayed on the sagittal plane ($x = 0$), rather creating elliptical paths 20 cm wide and forcing the robot paddles to move sideways as they tracked the ball. In extended experiments, the robot could maintain the ball on persistent oscillations for more than five minutes (more than 120 consecutive pushes).

The top row in Figure 5 shows a sequence of snapshots where the robot receives the ball from a person. On the same figure, the middle row shows the robot immediately pushing the ball back. An advantage of coupled phases is to provide a simple adaptive mechanism for the robot to compensate

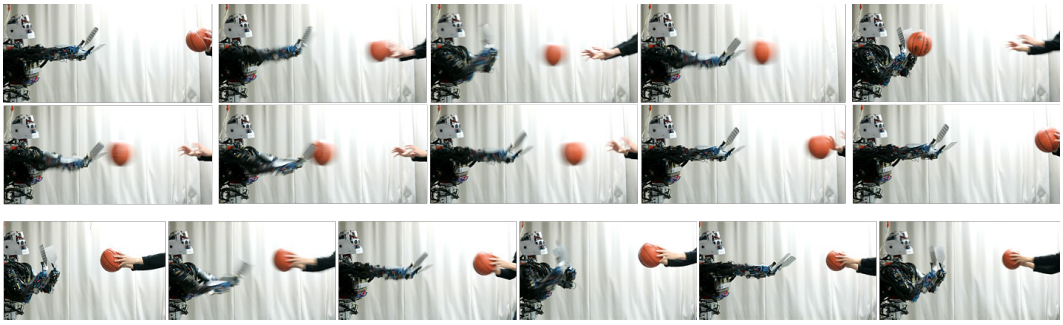

Figure 5: Execution of the bi-manual trained policy on the upper humanoid body. The upper row shows a sequence of snapshots where the robot receives the ball. The middle row shows the robot returning the ball right after. At the bottom row, the robot interacts with a person and it must be fast enough to handle situations in which the user grasps the ball and pretends a pass, but instead moves the ball sideways before making the final pass. (Refer to the accompanying video in http://www.cns.atr.jp/bri/en/robot/cb-i/ to see the whole experiment).

for temporal disturbances. As shown by the sequence of snapshots at the bottom row in Figure 5, the trained policy allows the robot to react to the ball movements even when the human tried to deceive it, by pretending to pass the ball, but actually moving it laterally. (See the accompanying video in http://www.cns.atr.jp/bri/en/robot/cb-i/ for a better understanding of the experiment).

Figure 6 shows disruptions and recoveries of the cycle as trajectories of the robot's left hand and the ball. The amplitudes of the curves are normalized to facilitate comprehension. At around 29.5 seconds, a person catches the ball and rocks it back and forth which leads the robot to react accordingly. She then releases the ball back to the robot at 30 seconds. The robot starts to recover the cycle when she jolts the ball towards the robot at around 33 seconds. Finally, the robot brings the ball back to a stable cycle at around 34 seconds.

4 Conclusion

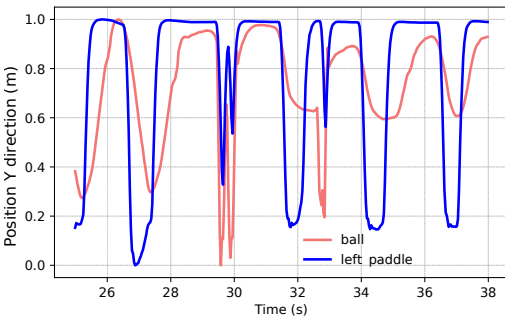

Figure 6: The horizontal components of the ball and the robot's left paddle trajectories projected on the sagittal plane of the robot. The spikes observed between 29.5-30 seconds are due to a person vigorously rocking the ball back and forth. At 33 seconds someone jolted the ball towards the robot. Note that the ball returns to a cyclic pattern whenever the disturbance is ceased.

This paper proposed a methodology to endow a robot with fast reactions to large spatio-temporal disturbances of a dynamic target. The phase-based formulation allows the progress of the robot to be directly driven by the moving target. To attain this goal, we revisited phase oscillators which have been mostly used in locomotion and investigated how to modulate the oscillator characteristics via policy search. To address spatial coordination, the method trains a sequence of joint distribution of robot poses and target locations whose indexes are given by the robot phase. The method was used on a humanoid upper body with 14 degrees-of-freedom interacting with a ball where the reward was designed to create a limit cycle. At times the ball was severely disturbed by someone reversing the ball trajectory and pretending passes, and the robot could gracefully adjust its phase and positions to recover and return the ball to a stable regime (refer to the accompanying video in http://www.cns.atr.jp/bri/en/robot/cb-i/).

We briefly illustrated our method in a single-stroke (discrete) case in the context of physical human-robot interaction. Our initial results in human-robot interaction showed that a reward that penalizes the timing between the human and robot hands is sufficient to swap the robot role from giver to receiver. Many exciting lines of future work are planned including locomotion where the moving target can be framed as planned step locations.

Acknowledgments

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO) and also supported by JSPS KAKENHI Grant Number JP16H06565.

References

[1] B. Bäuml, F. Schmidt, T. Wimböck, O. Birbach, A. Dietrich, M. Fuchs, W. Friedl, U. Frese, C. Borst, M. Grebenstein, et al. Catching flying balls and preparing coffee: Humanoid rollin'justin performs dynamic and sensitive tasks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3443–3444. IEEE, 2011.

[2] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson. Task-level robot learning: Juggling a tennis ball more accurately. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1290–1295. IEEE, 1989.

[3] W. Hong and J.-J. E. Slotine. Experiments in hand-eye coordination using active vision. In *Experimental Robotics IV*, pages 130–139. Springer, 1997.

[4] T. Kizaki and A. Namiki. Two ball juggling with high-speed hand-arm and high-speed vision system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1372–1377. IEEE, 2012.

[5] O. Koc, G. Maeda, and J. Peters. Online optimal trajectory generation for robot table tennis. *Robotics and Autonomous Systems*, 105:121–137, 2018. URL http://www.sciencedirect.com/science/article/pii/S0921889017306164.

[6] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[7] J. Kober, K. Mulling, O. Kromer, C. Lampert, B. Scholkopf, and J. Peters. Movement templates for learning of hitting and batting. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pages 853–858. IEEE, 2010.

[8] P. Kormushev, S. Calinon, and D. Caldwell. Robot motor skill coordination with EM-based reinforcement learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, 2010.

[9] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[10] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2): 79–91, 2004.

[11] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[12] N. Sugimoto and J. Morimoto. Phase-dependent trajectory optimization for cpg-based biped walking using path integral reinforcement learning. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 255–260. IEEE, 2011.

[13] F. H. Kong, A. M. Boudali, and I. R. Manchester. Phase-indexed ILC for control of underactuated walking robots. In *Control Applications (CCA), 2015 IEEE Conference on*, pages 1467–1472. IEEE, 2015.

[14] E. Klavins and D. E. Koditschek. Phase regulation of decentralized cyclic robotic systems. *The International Journal of Robotics Research*, 21(3):257–275, 2002.

[15] J. Morimoto, G. Endo, J. Nakanishi, S. Hyon, G. Cheng, D. Bentivegna, and C. G. Atkeson. Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1579–1584. IEEE, 2006.

[16] M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[17] J. Kober and J. R. Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009.

[18] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2397–2403. IEEE, 2010.

[19] F. Stulp, O. Sigaud, et al. Policy improvement methods: Between black-box optimization and episodic reinforcement learning. 2012.

[20] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen. Cb: A humanoid research platform for exploring neuroscience. *Advanced Robotics*, 21(10):1097–1114, 2007.

[21] E. Rohmer, S. P. N. Singh, and M. Freese. V-REP: a versatile and scalable robot simulation. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[22] G. Maeda, M. Ewerton, D. Koert, and J. Peters. Acquiring and generalizing the embodiment mapping from human observations to robot skills. *IEEE Robotics and Automation Letters*, 1 (2):784–791, July 2016. ISSN 2377-3766. doi:10.1109/LRA.2016.2525038.

Appendix: Simulating Training Data for Building Ball Pushing Spatial Models

As described in Section 2.1, the proposed movement primitive requires a sequence of joint distributions $P(\boldsymbol{q}, \boldsymbol{y})_k = \mathcal{N}(\boldsymbol{\mu}_{q,y}, \boldsymbol{\Sigma}_{q,y})_k$ and thus numerous training data. Rather than demonstrating multiple times the ball pushing task, we used a single human demonstration to collect the mean of the sequence of distributions, and then a heuristic to sample different poses along the mean demonstration. Figure 7(a) shows the demonstration where the demonstrator's left and right hand trajectories $\{\mathbf{x}_{human}^{demo}\}_{1:K}$ together with the motion of the ball being manipulated $\{\mathbf{x}_{ball}^{demo}\}_{1:K}$ were recorded with a marker tracking system. Using the method in [22], we retargeted the single expert demonstration onto the robot kinematics $\{\mathbf{x}_{human}^{demo}\}_{1:K} \rightarrow \boldsymbol{q}_{1:K}$ and assumed the demonstration to be the mean of the distribution.

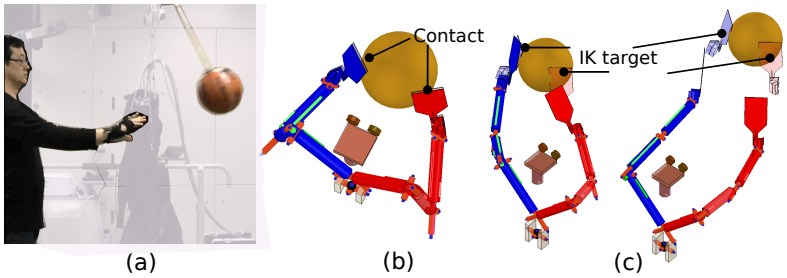

Figure 7: Learning the sequence of joint distributions from demonstration. The demonstration (a) mapped onto the robot kinematics was used to find the contact points between the robot paddles and the ball (b). We used these contact points as a heuristic to sample numerous ball positions along the demonstrated trajectory. The heuristic allowed paddle targets to be attached to the ball. As such, IK solutions could be computed for each sampled ball position (c), allowing us to train a rich joint distribution $P(\boldsymbol{q}, \boldsymbol{y}_{ball})$ without recurring to repeated human demonstrations.

Next, we artificially sampled $N = 50$ new pairs of ball and hand positions $(\boldsymbol{q}, \boldsymbol{y}_{ball})_n$ at each phase k along the demonstrated mean. We assumed that in a bi-manual ball pushing task there is a single, fixed ideal position of the hands on the surface of the ball at all times. As such, these

positions become evident when the ball is within the hands' reach, but it is only "virtually" maintained as the ball and hands depart from each other. Figure 7(b) shows the moment in which the robot paddles contact the ball. At this exact moment, we attached "paddle targets" on the ball surface at that specific position, and then sampled ball positions along the demonstrated trajectory $\{\boldsymbol{y}'_{ball} \sim \mathcal{N}(\mu = \boldsymbol{y}^{demo}_{ball}, \sigma^2)\}_{1:K}$. Since each ball sample has now the targets for the paddles, an IK solver was used such that the n-th training pair $(\boldsymbol{q}', \boldsymbol{y}'_{ball})_n$ could be obtained from $\boldsymbol{q}' = \text{IK}(\boldsymbol{y}'_{ball})$. Figure 7(c) shows two such instances. Once the samples were obtained, the procedure described in Section 2.1 was subsequently used.